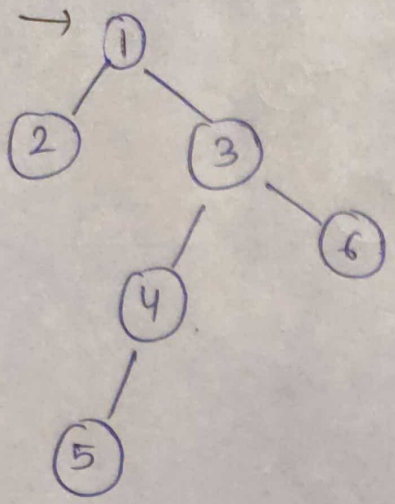
Maximum Depth Of Binary Tree:-



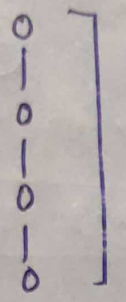
Recursive / Level order.

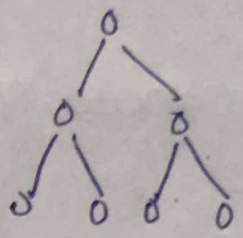
↓

It would take O(height) Auxiliary space.

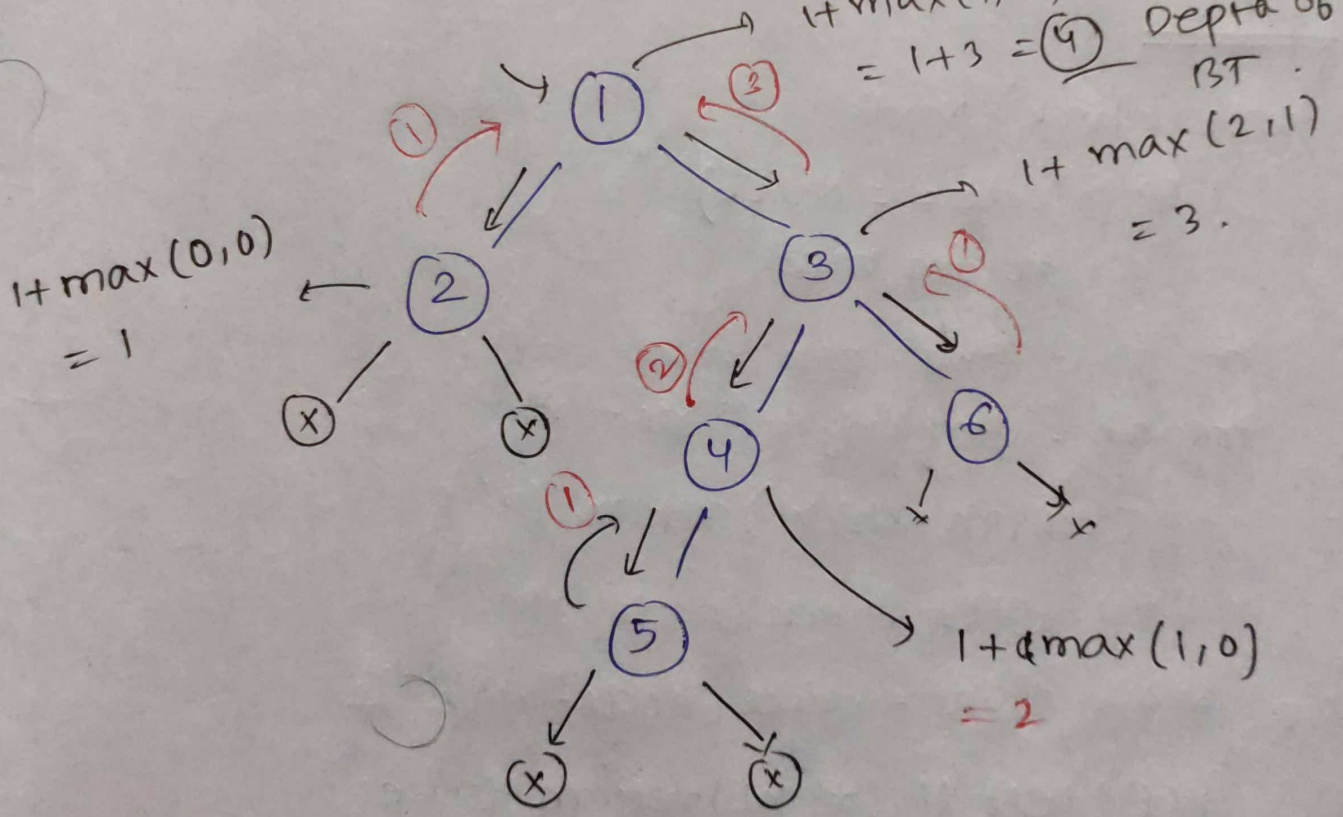
↓

It would take O(n) time if there is complete BT. & last level is completely filled.





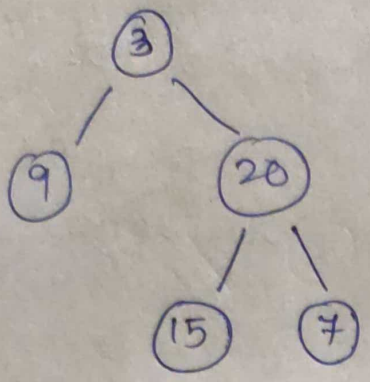
1 + max(l, r)

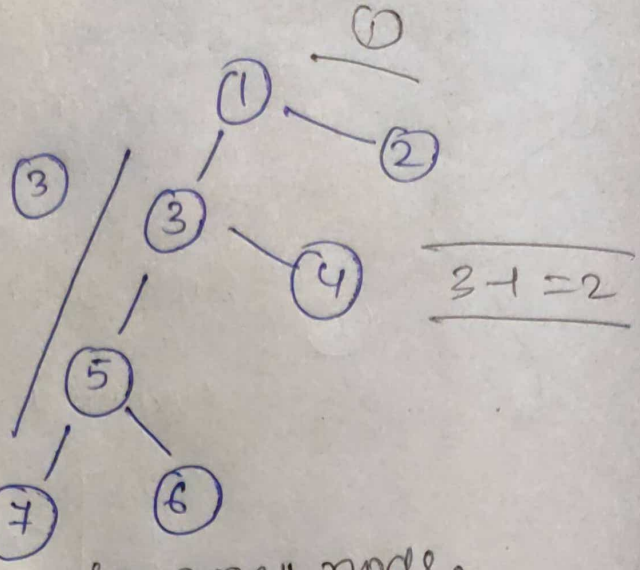


Code :-

```
int maxDepth(TreeNode *root) {
    if (root == NULL) return 0;
    int lh = maxDepth(root->left);
    int rh = maxDepth(root->right);
    return 1 + max(lh, rh);
}
```

Check for Balanced Binary Tree:-





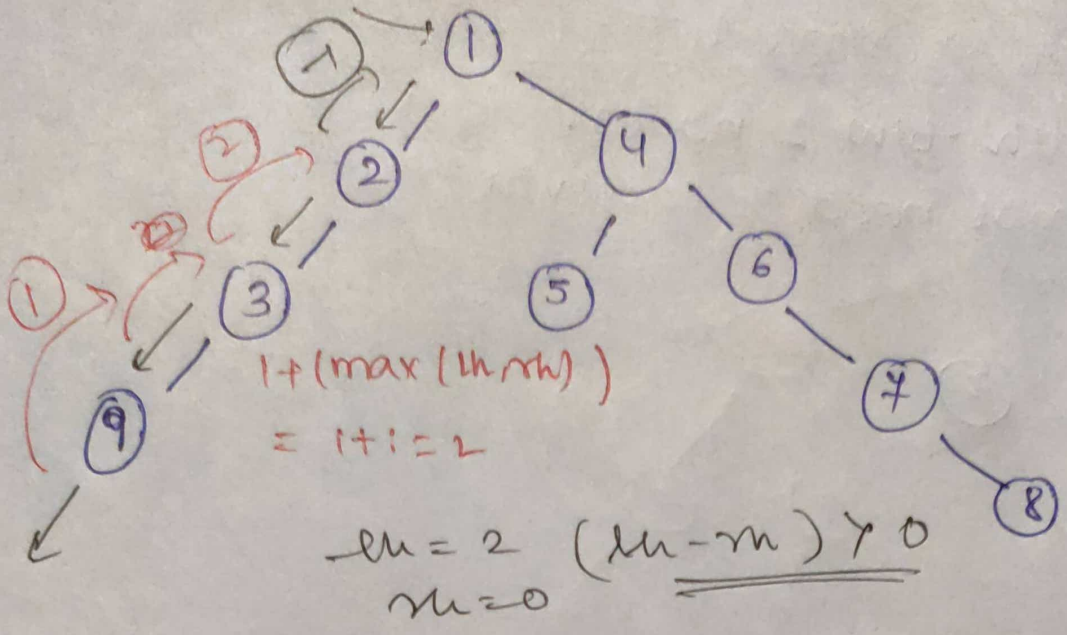
Balanced Binary Tree → for every node,

height(left) - height(right) <= 1

Naive:-

```
bool check (Node)
    if (node == NULL)
        return true;

    int lh = findHeight (node -> left);
    int rh = findHeight (node -> right);

    if (abs (rh - lh) > 1) return false;

    bool left = check (node -> left);
    bool right = check (node -> right);

    if ( ! left || ! right) return false;
    return true;
```

$O(N \times N) = O(N^2)$




```
bool isBalanced (TreeNode *right) {
    return dfsHeight (root) != -1;
}
int dfsHeight (TreeNode *root) {
    if (root == NULL) return 0;
    int lh = dfsHeight (root -> left);
    if (leftHeight == -1) return -1;
    int rh = dfsHeight (root -> right);
    if (rightHeight == -1) return -1;
    if (abs (lh - rh) > 1) return -1;
    return max (lh, rh) + 1;
}
```